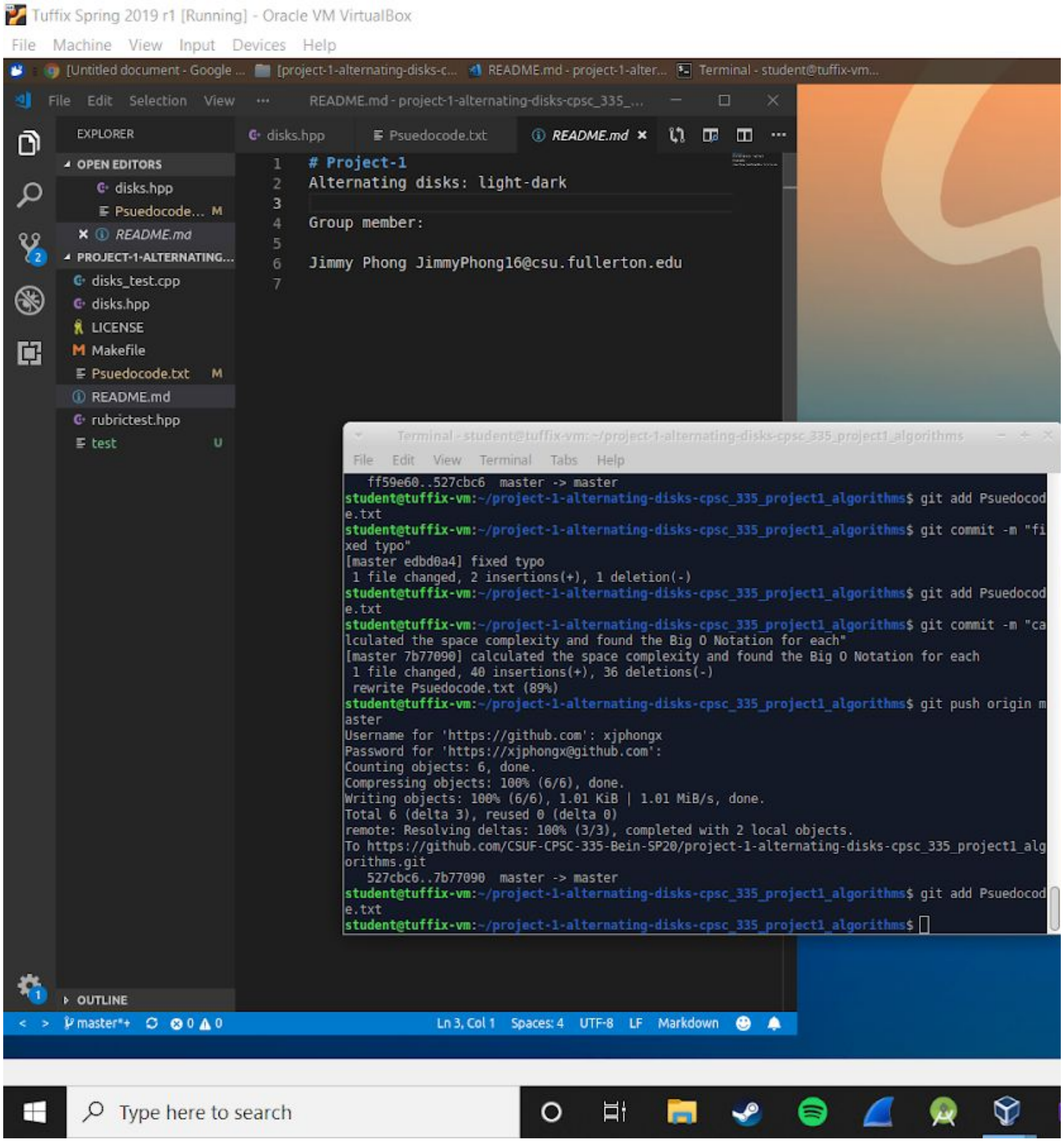
CPSC 335 Project 1 - Analyzing Algorithms

By: *Jimmy Phong*
JimmyPhong16@csu.fullerton.edu

In this project, I was given the task to analyze two algorithms: Left-to-Right Algorithm and Lawnmower Algorithm. This project description includes complete functionality of the running skeleton code provided by our professor, formatility of the report and github presentation, and a complete analysis of the algorithm's efficiency class. Everything is up-to-date and uploaded on to my Github repository.

Pasted below is my screenshot of Tuffix and the IDE that I used, which is Visual Studio Code.


Tuffix Spring 2019 r1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
[Untitled document - Google ...
[project-1-alternating-disks-c...
README.md - project-1-alter...
Terminal - student@tuffix-vm...
File Edit Selection View ···
README.md - project-1-alternating-disks-cpsc_335_...
EXPLORER
disks.hpp
Psuedocode.txt
README.md
OPEN EDITORS
disks.hpp
Psuedocode... M
README.md
PROJECT-1-ALTERNATING...
disks_test.cpp
disks.hpp
LICENSE
Makefile
Psuedocode.txt M
README.md
rubrictest.hpp
test U
OUTLINE
master*+
0 0
Ln 3, Col 1
Spaces: 4
UTF-8
LF
Markdown
Type here to search

```
# Project-1
Alternating disks: light-dark

Group member:

Jimmy Phong JimmyPhong16@csu.fullerton.edu
```

Terminal - student@tuffix-vm: ~/project-1-alternating-disks-cpsc_335_project1_algorithms
File Edit View Terminal Tabs Help

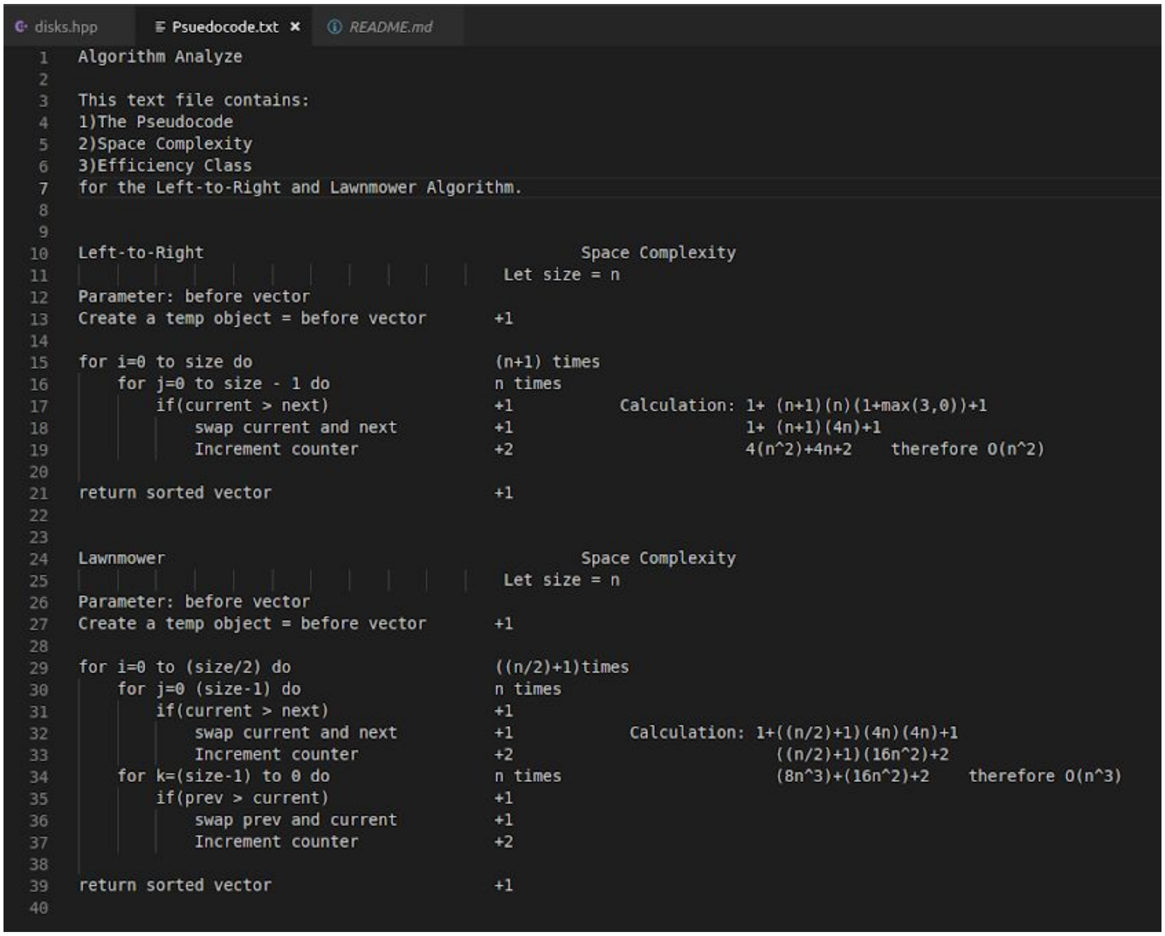
```
   ff59e60..527cbc6  master -> master
student@tuffix-vm:~/project-1-alternating-disks-cpsc_335_project1_algorithms$ git add Psuedocod
e.txt
student@tuffix-vm:~/project-1-alternating-disks-cpsc_335_project1_algorithms$ git commit -m "fi
xed typo"
[master edbd0a4] fixed typo
 1 file changed, 2 insertions(+), 1 deletion(-)
student@tuffix-vm:~/project-1-alternating-disks-cpsc_335_project1_algorithms$ git add Psuedocod
e.txt
student@tuffix-vm:~/project-1-alternating-disks-cpsc_335_project1_algorithms$ git commit -m "ca
lculated the space complexity and found the Big O Notation for each"
[master 7b77090] calculated the space complexity and found the Big O Notation for each
 1 file changed, 40 insertions(+), 36 deletions(-)
 rewrite Psuedocode.txt (89%)
student@tuffix-vm:~/project-1-alternating-disks-cpsc_335_project1_algorithms$ git push origin m
aster
Username for 'https://github.com': xjphongx
Password for 'https://xjphongx@github.com':
Counting objects: 6, done.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.01 KiB | 1.01 MiB/s, done.
Total 6 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To https://github.com/CSUF-CPSC-335-Bein-SP20/project-1-alternating-disks-cpsc_335_project1_alg
orithms.git
   527cbc6..7b77090  master -> master
student@tuffix-vm:~/project-1-alternating-disks-cpsc_335_project1_algorithms$ git add Psuedocod
e.txt
student@tuffix-vm:~/project-1-alternating-disks-cpsc_335_project1_algorithms$ 
```

Pseudo Code Listing

```
Left-to-Right

Parameter: before vector
Create a temp object = before vector

for i=0 to size do
   for j=0 to size - 1 do
       if(current > next)
           swap current and next
           Increment counter

return sorted vector



Lawnmower

Parameter: before vector
Create a temp object = before vector


for i=0 to (size/2) do
   for j=0 (size-1) do
       if(current > next)
           swap current and next
           Increment counter
   for k=(size-1) to 0 do
       if(prev > current)
           swap prev and current
           Increment counter
```

Algorithm Analysis

This is also included in my github repository.



```
Algorithm Analyze

This text file contains:
1)The Pseudocode
2)Space Complexity
3)Efficiency Class
for the Left-to-Right and Lawnmower Algorithm.


Left-to-Right                                       Space Complexity
                                            Let size = n
Parameter: before vector
Create a temp object = before vector       +1

for i=0 to size do                         (n+1) times
    for j=0 to size - 1 do                 n times
        if(current > next)                 +1          Calculation: 1+ (n+1)(n)(1+max(3,0))+1
            swap current and next          +1                       1+ (n+1)(4n)+1
            Increment counter              +2                       4(n^2)+4n+2    therefore O(n^2)

return sorted vector                       +1


Lawnmower                                           Space Complexity
                                            Let size = n
Parameter: before vector
Create a temp object = before vector       +1

for i=0 to (size/2) do                     ((n/2)+1)times
    for j=0 (size-1) do                    n times
        if(current > next)                 +1
            swap current and next          +1           Calculation: 1+((n/2)+1)(4n)(4n)+1
            Increment counter              +2                          ((n/2)+1)(16n^2)+2
    for k=(size-1) to 0 do                 n times                     (8n^3)+(16n^2)+2    therefore O(n^3)
        if(prev > current)                 +1
            swap prev and current          +1
            Increment counter              +2

return sorted vector                       +1
```

Analysis Report

After analyzing the space complexity for both algorithms, I have come to the conclusion that the Lawnmower algorithm is more efficient than the Left-to-Right(LR) algorithm. According to the calculations presented above, the space complexity for LR came out to becoming (4n^2)+4n+2 which has the efficiency of O(n^2). On the other hand, The lawnmower algorithm has the space complexity of (8n^3)+(16n^2)+2 which has the efficiency of O(n^3). As n inputs goes into both LR and Lawnmower algorithm, Lawnmower algorithm is more efficient than LR because the graph of n^3 is higher than the graph of n^2.